AN APPROXIMATION ALGORITHM WITH ADDITIVE ERROR FOR EXTENSIVE-FORM, PURE STACKELBERG GAMES WITH A CHANCE PLAYER

by

Matthew Allbee

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

(Computer Science)

at the

UNIVERSITY OF WISCONSIN–WHITEWATER

November, 2018

Graduate Studies

The members of the Committee approve the thesis of

Matthew Allbee presented on 30 November 2018

(Dr. Jiazhen Zhou)

(Dr.Arnab Ganguly)

(Dr. Athula Gunawardena)

AN APPROXIMATION ALGORITHM WITH ADDITIVE ERROR FOR EXTENSIVE-FORM, PURE STACKELBERG GAMES WITH A CHANCE PLAYER

By

Matthew Allbee

Under the supervision of Assistant Professor Jiazhen Zhou

At the University of Wisconsin-Whitewater

Substantial work has gone into finding techniques for solving real-world sized Nash games. Stackelberg Equilibria is another important solution concept for game theory models relevant to Computer Science, but much less progress has been made for solving very large Stackelberg games. This thesis is a step in that direction, presenting an algorithm which can find an $OPT-\epsilon$ approximate solution to the NP-Hard problem of perfect information, extensive-form, pure-strategy Stackelberg games with chance nodes in $O(b\epsilon^{-2}|V|)$; where b is the maximum branching factor of the game tree and $|V|$ is the number of nodes in the tree.

ϵ-Reachability is compared both theoretically and through simulations to a previous folly-polynomial approximation algorithm, which we'll refer to as $FPTAS$. $FPTAS$ runs in $O(b(\frac{|h_\emptyset|}{\epsilon})^2)|V|)$, where $|h_\emptyset|$ is the height of the game tree. In simulations, this extra $|h_\emptyset|$ factor proves an extremely limiting variable in relation to the run-time of ϵ-Reachability for even moderate values of $|h_\emptyset|$. This is likely to prevent $FPTAS$ from being feasible on tall game trees - a likely feature of real-world extensive-form games. As both algorithms are linear in non-chance and quadratic in chance nodes, they both are sensitive to the frequency of chance nodes. Again, though, $FPTAS$'s dependence on $|h_\emptyset|$ is likely to cause its run-time to grow much more quickly than ϵ-Reachability as the proportion of chance nodes increases.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

Introduction

With computers increasingly being assimilated into the structure of society, it has become apparent that we must consider not only how technology behaves but also the interaction between users and technology. Many use cases of technology require the cooperation of multiple users - users who may not necessarily wish to cooperate. In order to make these use cases workable, we must develop methods for incentivizing users to cooperate.

Games are a frequent tool used in modelling strategic iteractions between at least 2 self-interested agents. In a game, each of a set of players must choose between a set of actions in a way that maximizes their reward. The reward for each player is not determined solely by the player's action, but by the interaction of the actions taken by all players.

In Computer Science, games have become an important model for studying a variety of problems. These include studying congestion characteristics of different network topologies, designing optimal combinatorial auctions, and designing communication protocols. Especially important to this thesis are *security games*, in which an attacker attempts to steal or destroy some set of resources while a defender attempts to protect them. Security games are used to models in computer security, as well the protection of endangered wildlife and military targets.

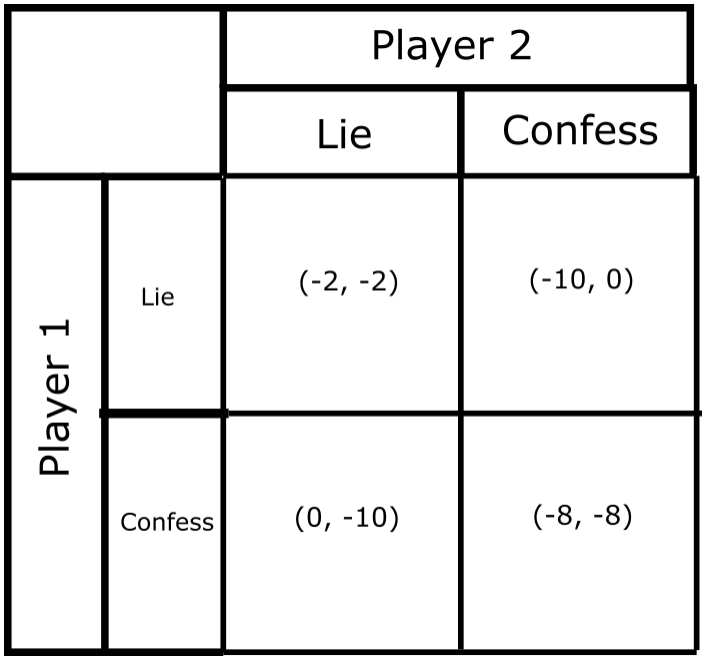


Figure 1.1 Prisoners' Dilemma Game

1.1 Normal-Form Games

Normal-form games model one-off situations in which players take a single action before receiving a reward. We will only consider games with finite players and for which each player has a finite set of actions. These games have nice theoretical properties, that will be discussed in later sections.

Two-player normal-form games are often represented in *matrix form*. Below is the often discussed Prisoners' Dilemma game in normal form:

The prisoner's dilemma models a situation in which two criminals are being interrogated about a crime. The police have evidence enough to jail both criminals for 2 years, but they know they believe that they had also committed a more serious crime. To force a confession, they present each criminal with a proposition: confess that both he and the other criminal had

committed to a crime and avoid jail time. If only one criminal confesses, though, the non-confessing criminal gets 10 years of jail. If both confess, they will both be sentenced to 8 years in jail.

In the diagram above,"lie" and "confess" represent the criminals' possible actions. Within each box, the tuple of numbers represent each players possible "rewards"; with the first number in the tuple being the first criminal's reward.

More formally, a normal-form game $G :=< P, A, \mu >$ is a tuple of players P, possible action sets A, and reward functions μ. For each player p_i, and player specific variable v, we let v_i refer to p_i's instance of v. Also $v_{-i} := \times_{p_j \in P \setminus \{p_i\}} v_i$. In this vein, each player $p_i \in P$ chooses some action of their possible actions $\Pi_i \subset A$. Each $\sigma_i \in \Pi_i$ is a pure strategy of player i and Π_i is i's *pure strategy set*. If players are allowed to randomize over their strategies, we denote all such randomized strategies by Δ_i; player i's *mixed strategy set*. For each player p_i, their is a reward function $\mu_i : \times_{p_i \in P}(A_i) \longrightarrow \mathcal{N}$.

1.2 Equilibrium Concepts

The structure of games do not inherently lead to a method for deciding how players will choose their actions - which strategy each player will settle on. To determine this, we may turn to various *equilibrium concepts*. An equilibrium is some formula describing how each player will choose from strategy, dependent on how all other players play.

1.2.1 Nash Equilibrium

The most popular of these is the *Nash Equilibrium*. Nash Equilibria model competitive scenarios in which collusion between players is infeasible or unhelpful. They are defined for both cases in which players can play pure and mixed strategies. A *Pure Nash Equilibrium* is defined by a choice of some pure strategy σ_i by each player p_i such that:

$$\forall \sigma_i' \in \Pi_i;\ \mu_i(\sigma_i, \sigma_{-i}) \geq \mu_i(\sigma_i', \sigma_{-i})$$

where σ_{-i} is the choice of pure strategies by all other players besides p_i. A *Mixed Nash Equilibrium* is defined similarly, but replacing pure strategy sets Π_i with mixed strategy sets Δ_i for each player.

1.2.2 Correlated Equilibrium

Another popular equilibrium concept is the *Correlated Equilibrium*. These form a superset of Nash Equilibria, but are guaranteed even for games with an infinite player set, in which each player may have an infinitely-sized pure strategy set [12]. This solution concept deals with scenarios in which players may coordinate around some public signal - such as may be provided by a trusted third party. For instance, a common game used in the discussion of Correlated Equilibria is *Chicken*:

This game models a contest in which two drivers are dared to drive their cars directly at one another. The first person to "chicken out" and pull off course will be seen as less courageous, and so will receive a lesser reward than the person who never breaks from the dare. If neither chickens out, though, the cars crash into each other. Nobody is a winner in that situation and so neither driver receives a reward.

Now, imagine there is a trusted third party, committed to making both drivers look good, who indicates to each driver (separately) which should chicken out in each round. After considering how best to do this, the third party settles on the following indications to the two players:

The third party will signal chicken to player 1 and dare to player 2 1/3 of the time, will signal dare to player 1 and chicken to player 2 1/3 of the time, and chicken to both players 1/3 of the time.

Now, note that both players cannot increase their reward by deviating from these suggestions. When it is indicated that player 1 chicken out, they know that, if player 2 follows their recommendation, player 1's expected reward is $\frac{1}{2} * 2 + \frac{1}{2} * 6 = 4$. This is greater than the expected reward from deviating to "dare," $\frac{1}{2} * 0 + \frac{1}{2} * 7 = 3.5$. Likewise, when the third

Figure 1.2 Chicken Game

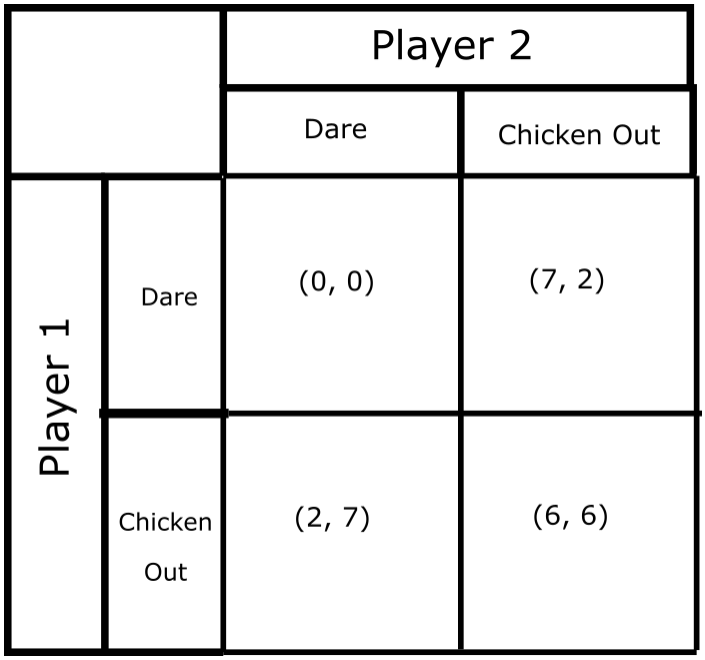


Figure 1.3 Public Signal

Trusted Party's Suggestions			
Player 1	Dare	Chicken	Chicken
Player 2	Chicken	Dare	Chicken
Probability	1/3	1/3	1/3

party suggests that player 1 "dare," the expected reward is 7, greater than the 6 for deviating to chicken out. By symmetry, player 2 will also be better off by listening to the third party's recommendations, and so both players will choose to follow their suggested strategy at all times - leading to the Correlated Equilibrium.

1.2.3 Stackelberg Equilibrium

Another equilibrium concept, of course, is the Stackelberg Equilibrium. This is used to model situations in which one player has a first-move advantage - either acting or committing to some actions before the other player(s) are able to act. In Stackelberg games, one player has a special *leader* designation, which designates that this player selects their strategy before the other players. The other *follower* players then choose a best response to the leader's strategy. Most commonly, Stackelberg games have only two players: a single leader and a single follower.

In pure strategies, a Stackelberg Equilbrium is a solution to the following equation:

$$\underset{\sigma_1 \in \Pi_1; \sigma_2 \in BR(\sigma_1)}{\arg\max} \mu_1(\sigma_1, \sigma_2)$$

where $BR(\sigma_1)$ is the set of the follower's best response strategies to σ_1. That only the follower must choose a strategy which is a best response is in contrast to the Nash Equilibrium; in which each player must choose a strategy which is a best response to every other player's strategy. This additional autonomy may allow the leader to increase their reward above that which can be achieved by a best response. The game in figure 1.4 portrays a specific instance in which this occurs.

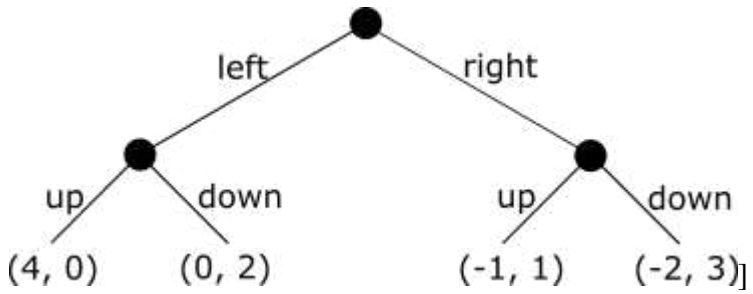
Note that this game contains a single, pure Nash Equilibrium, of course also a mixed strategy, in which the follower plays up and the leader plays right. The leader's reward, though, is not the largest reward that the leader can achieve under a Stackelberg Equilibrium. To see this,

Figure 1.4 Stackelberg Game

		Leader	
		Left	Right
Follower	Up	(4, 0)	(-1, 1)
	Down	(0, 2)	(-2, 3)

consider the tree representation of the game, in which the leader selects their action before the follower:

Figure 1.5 Stackelberg Representation of Normal-Form Game



If the leader commits to action "left," then the follower's best response is the "down" action. This results in the leader getting 0 reward, better than the -1 reward of the sole Nash Equilibrium.

The Stackelberg Equilibrium was first defined, and much of the early work is based on, the Stackelberg competition. In these games, two firms sell an identical product, and must determine what quantity to produce in order to maximize profits. One of these firms is a leader, though, and so sets their production before the follower player.

In Computer Science, the major use of Stackelberg Equilibria is in a *Security Game*. These games model the competition between a defender and attacker of some set of resources. In the

general model, the defender and attacker both have a set of ”units,” often meant to represent defensive/attack personnel, which they must allocate to the resources. The attacker wishes to allocate its units to resources which haven't been allocated by the defender, while the defender wants the opposite. If the defender sets their defensive policy before the attacker may act, modelled by a Stackelberg Equilibrium, then we have a *Stackelberg Security Game*.

More formally, a security game is a game with a defender player and attacker player, each of which has a set of units, and a set of resources. Both players must choose a probability distribution over their possible unit allocations. The defender seeks to maximize the expected number of resources in which both have allocated a unit, while the attacker seeks to minimize this.

This general model does not capture the wide variety of security game variants, though. For example, patrol security games involves the “attacker's” attempt to traverse a graph while avoiding ever sharing a node simultaneously with the defender.

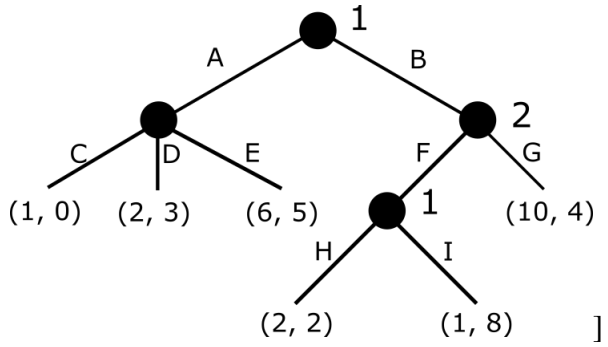
Stackelberg Security games have a number of different applications relevant to Computer Science. Stackelberg Security games have been used to model vulnerabilities in offloading of tasks to the cloud [26]. They have also been used to study eavesdropping/jamming in wireless networks [19].

1.3 Extensive-Form Games

In contrast to normal-form games, ***extensive-form games*** model situations in which players face a sequence of action choices. These games are often represented with *game trees*. An example is given below:

This game begins with player 1 choosing between two actions, A and B. Depending on which action player 1 chooses, player 2 may either have to choose between C, D, and E or F and G. Finally, if player 2 has chosen action F, player 1 must choose between the H and I actions. When one of the tree's terminal nodes is reached the game ends and both players recieve some reward. This reward is represented by the pair of numbers at each terminal node;

Figure 1.6 Extensive-Form Game



]

with the first number being player 1's reward at that terminal node and the second being player 2's reward. Finally, next to the rightmost node in each level of the tree is printed the index of the player who acts at that level of the tree. an example, if player 1 plays action A and player 2 then plays action D, then players 1 and 2 will receive respectively rewards of 2 and 3.

1.3.1 Equilibrium in Extensive-Form Games

Equilibrium concepts are identified identically in extensive-form and normal-form games. In fact, one can transform extensive-form games into normal form; although this may cause exponential blow-up in the size of the representation. The strategies which player's may use, though, involve not just a single action, but must account for all nodes in the game tree at which the player acts.

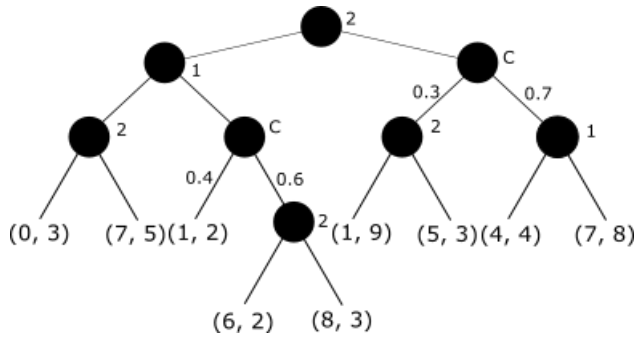
For a Stackelberg Equilibrium in an extensive-form game, the leader "commits" to not just a single action, but to the action which they will take from each node in the tree at which they act. Likewise, the follower chooses a best response from every node in the tree at which they act.

1.3.2 Extensive-Form Games with a Chance Player

We will be considering extensive-form games with a chance player. A chance player is a method for modelling probabilistic aspects in a game. At each node in which the chance player

acts, they will choose their actions according to some pre-defined probability distribution. Below is an example of a game tree with a chance player.

Figure 1.7 Extensive Form Game with a Chance Player



The figure is similar to the previous game tree example. Nodes at which the chance player acts are designated by a C, and the numbers next to the edges emanating from these nodes indicate with the probability that the chance player will select the action leading down that edge.

1.4 Contributions

Very little work has gone into finding strong solutions for real-world sized extensive-form Stackelberg games. This thesis attempts to make progress towards this with the goal of creating an algorithm which finds approximates a Stackelberg leader strategy quickly enough to be feasible to use on these very large cases. Further, the hope is that the algorithm degrades gracefully, allowing reasonable increases in the size of ϵ as the size of the game grows. To this aim, we will design such an algorithm, which will run in $O(|V|b\epsilon^{-2})$; where $|V|$ is the number of nodes in the game tree and b is the tree's maximum branching factor. Simulations are performed to show that the algorithm runs more quickly, and can handle larger game trees, than the only other approximation algorithm which could be found in the literature to the knowledge of this author.

Chapter 2

Related Work

With the advent of the internet, game theory has also become an important tool in Computer Science [21]. Much of this work has gone into the study of combinatorial auctions [9], in which bidders are given a set of items and can submit bids for subsets of those items. This format has been used to allocate radio spectrum allocation, employee vacation time, and truck delivery routes [9].

Games have also found widespread use in the study of network congestion [18], leading to advances in algorithms for fair bandwidth sharing [11], and congestion-based pricing models [2]. More broadly, games have been used to model mesh networks [5] and to maximize user-perceived quality-of-service (QoS) in network control flow [14].

Another area of wide application is in job scheduling [13], treating the task of job scheduling as a game in which a number of players is assigned some jobs and must use some shared resources to complete them. A problem with a wide variety of applications, games have been developed to study a variety of different instances. Some instances of these include grid computing with selfish agents [16], multi-processor scheduling [1], job scheduling on un-related machines [6], and scheduling of jobs in networking manufacturing [27].

Of particular importance when discussing Stackelberg Equilbria are *security games*. These games, usually with only 2 players, model situations in which one player, the attacker, seeks to destroy or steal a set of resources while the other player, the defender, must act to defend them. Some major applications of security games have been modelling the tactics of poachers [10], designing optimal security routes for airport security [22], optimal placement of air marshals and other security personnel in transportation networks [24], and defending against false

information spread in social networks [20]. There are also a number of importance varieties of security games.

Finding a Stackelberg Equilibrium in extensive-form games is NP-Complete for many cases, including in pure strategies with a chance player [3]. Mixed integer linear programs have been put forward to find (mostly mixed) Stackelberg Equilibria in extensive-form games [4]. Little work has been done in approximating Stackelberg Equilibria in games too large to be solved this way.

Finding strong strategies in very-large, extemsive-form games under Nash Equilibria has been studied extensively; espeically in the case of imperfect information about the other players past moves. One set of techniques focuses on game abstractions; in which nodes are pruned from the game tree. A number of abstraction techniques have been put forward. Many of these techniques can even bound the loss to the player from only considering part of the tree, leading to a sort of approximation algorithm [15]. Abstractions, though, may behave pathologically [25]. Nash Equilibria strategies in larger abstractions may have less error in relation to an Equilibrium in the un-abstracted game than those found in those with fewer nodes removed. Still, abstractions have proven themselves useful in the real world. Node pruning was used in the AlphaGo model which was the first to defeat a human professional Go player [23].

These techniques have had to be pursued in part because of the PPAD-Completeness of approximating a Nash Equilbrium with either multiplicative or additive error [8]. It may still be, though, that is is possible to compute an approximation, within some constant factor, of a Stackelberg Equilibrium quickly enough to be relevant to large games. For many cases, it has been proven that an approximate Stackelberg Equilibrium can be found in polynomial time on extensive-form games [3]. Of these, both pure and mixed Stackelberg Equilibria can be approximated with additive error in polynomial time for extensive-form games with a chance player.

There has also been some work in randomized algorithms for finding Stackelberg Equilibria. For instance, [7] described an algorithm which would find a randomized pure strategy for the follower and showed that this results in approximately a best response from the follower.

Chapter 3

Theoretical Work

Games are formalized in *sequential form*. This formalizes each node as a *history*. Each history is, simply, a sequence of the previous actions which were taken to reach that node/history. Expectedly, edges in the game tree are are formalized as *actions*. At each history, some player may choose from some history-specific set of actions. The player's choice determines which history will come next in the game.

3.1 Game Framework

Let $\mathcal{G}$ be set of all games and $G \in \mathcal{G}$ be some specific game. Then we will define G to be the tuple $< N, H, Z, A, \rho, \mu, \mathcal{C}, \mathcal{I} >$, where:

- $N := \{p_1, p_2, p_c\}$: The set of players. Throughout, we assume that player 1 (p_1) is the leader and player 2 (p_2) is the follower in relation to a Stackelberg Equilibrium.
- $p_c \in N$: A chance player who chooses probabilistically between actions in games with chance nodes
- A: The set of all actions which may be playable in G.
- H: The set of possible play histories; each history corresponds to some finite sequence of actions.
- $h_\emptyset \in H$: The *root history* from which the game begins. $h_\emptyset$ is equivalent to the empty sequence of actions.

- $Z \subseteq H$: The set of *terminal histories* - at which the game terminates and each player receives some reward
- $\forall p_i \in N; \mu_i : Z \to [0, 1]$: The set of *reward functions* assigning a reward between 0 and 1 inclusive to each player p_i at each terminal history.
- $\rho : H \to N$: maps histories to the player which acts when that history has been reached.
- $act : H \to 2^A$ Is the *action function*, mapping each history to a set of actions which may be taken at that history.
- For some $h \in H$ with $\rho(h) = p_i$, let $a \in act(h)$ be a possible action of player p_i at h.
- $ha = h' \in H$ is the history reached by playing the action a at h.
  - h' is the sequence created by appending a to the end of h.
- $\chi : H \to 2^H$: The *children function*, defines the set of histories which can be reached from each $h \in H$ by some single action
- $par : H \to H$: The *parent function*, defines the *parent history*, for which a given history is a child

- $\mathcal{C} : A \to (0, 1]$: The probabilistic *chance function*, It defines, for any $h \in H$ where $\rho(h) = p_c$, the (strictly positive) probability that p_c will choose each action $a \in act(h)$. Note that $\Sigma_{a \in act(h)} \mathcal{C}(a) = 1$.

For some $h \in H$, define the **height** of h to be the length of the shortest sequence of actions from h to some terminal history. Denote this by $|h|$ (i.e., if $|h| = 0$, then h is a terminal history).

Also, in general, for some history h, let v_h be the value of some variable v constrained to the subgame rooted at h. For example, Z_h is the set of terminal histories in the subtree rooted at h.

3.2 Player Strategies

Each player has a set of *strategies*, corresponding to the different ways that the player can act at each relevant history. For player p_i, this strategy set is denoted by Π_i, where the strategy of player p_i is some $\sigma_i \in \Pi_i$.

The entire sequence of play is then determined by the *strategy pair* $(\sigma_1, \sigma_2) \in \Pi_1 \times \Pi_2$, where σ_1 is the strategy chosen by player 1 and σ_2 is the strategy chosen by player 2, along with the actions chosen by the chance player.

In a Stackelberg Equilibrium, p_2 (the "follower") will select their strategy while knowing how player 1 will act. So, it is natural to expect that they will constrain their strategy choice to some *best response* to player 1's choice. Here, a best response is a choice of strategy which maximizes player 2's reward given some strategy for player 1. For some strategy σ_1 by player 1, denote player 2's best response strategies by $BR(\sigma_1)$.

This leads to the idea of a *feasible strategy pair*.

Feasible Strategy Pair : A strategy pair (σ_1, σ_2) is feasible iff $\sigma_2 \in BR(\sigma_1)$.

3.3 Stackelberg Equilibrium

A Stackelberg Equilibrium models (2-player) strategic situations in which player 1 commits to and communicates their strategy to player 2 before player 2 can choose their strategy. In economic contexts, this is often used to model a firm's choice to enter a market after witnessing that a competitor has entered the market. In computer science, Stackelberg games are primarily used to model security scenarios in which an attacker of some system can determine the actions of the defender before beginning their attack.

Formally, a Stackelberg Equilibrium is a strategy pair (σ_1, σ_2) which satisfies:

$$(\sigma_1, \sigma_2) = \underset{\sigma_1' \in \Pi_i; \sigma_2' \in BR(\sigma_1')}{\arg\max} \mu_1(\sigma_1', \sigma_2')$$

Where $\mu_1(\sigma'_1, \sigma'_2)$ is the reward which player 1 receives when player plays according to strategy σ'_1 and player 2 plays accordig to strategy $'_2$.

In chance games, both $BR(\sigma_1)$ and $\mu_1(\sigma_1, \sigma_2)$ are not well-defined as the terminal history reached is dependent on the probabilistic actions of the chance player. To remedy this, we will consider a Stackelberg Equilibrium to be a strategy pair which satisfies:

$$\sigma_1, \sigma_2 = \underset{\sigma'_1 \in \Pi_1; \sigma_2 \in \mathbb{E}[BR(\sigma'_1)]}{\arg\max} \mathbb{E}[\mu_1(\sigma_1, \sigma_2)]$$

where $\mathbb{E}[BR(\sigma_1)]$ is the set of responses by p_2 with the maximum expected value of μ_2 when p_1 plays σ_1 and $\mathbb{E}[\mu_1(\sigma_1, \sigma_2)]$ is the expected reward to p_1 when p_1 plays σ_1 and p_2 plays σ_2. Also, we will substitute $\mathbb{E}[BR(\sigma_1)]$ in place of $BR(\sigma_1)$ in the definition of feasible strategy pair.

3.4 Finding a Stackelberg Equilibrium in Chance Games

(Probabilistic) Reachability: A terminal history $z \in Z$ is (probabilistically) reachable iff there is some feasible strategy pair (σ_1, σ_2) such that the game terminates at z with positive probability.

Determining the *reachability* of terminal histories is a vital part to the regret-bounding algorithm.

Under a strategy pair (σ_1, σ_2), there may be multiple reachable terminal histories, corresponding to the possible actions of the chance player p_c. Define the set of reachable terminal histories under (σ_1, σ_2) as $Z(\sigma_1, \sigma_2)$. Also, for each strategy pair, associate a function $\pi(\sigma_1, \sigma_2)[z]$ to be the probability of reaching $z \in Z$ when the (σ_1, σ_2) strategy pair is played. Thus, $\pi(\sigma_1, \sigma_2)[z] > 0$ iff $z \in Z(\sigma_1, \sigma_2)$. Finally, define $< Z(\sigma_1, \sigma_2), \pi(\sigma_1, \sigma_2) >$ to be the *terminal history distribution* under (σ_1, σ_2). We will refer to both $Z(\sigma_1, \sigma_2)$ as reachable as shorthand to mean that all terminal histories in $Z(\sigma_{1,2})$ are reachable.

We may also refer to the terminal history distribution $< Z(\sigma_1^c, \sigma_2^c), \pi(\sigma_1^c, \sigma_2^c) >$ for a strategy pair (σ_1^c, σ_2^c) which is only defined from some child c, such as a strategy pair defined in

the subgame G_c, as if it were also defined at the parent h of c. In these instances, assume that (σ_1^c, σ_2^c) is extended to h so that the action taken at h leads to c.

We will begin by defining an algorithm, **Reachability**, which is an adaptation of the algorithm described in [17] for games without chance. Reachability determines $< Z(\sigma_1, \sigma_2), \pi(\sigma_1, \sigma_2) >$ for all feasible strategy pairs (σ_1, σ_2). As finding a Stackelberg Equilibrium in pure-strategy chance games has been shown to be NP-Complete, and we can easily compute a Stackelberg Equilibrium with this information by simply selecting the reachable terminal history distribution with greatest expected reward to player 1, this algorithm takes exponential time in the size of the game tree. Later, our approximation algorithm will approximate this while running in poly-time.

Reachability is defined recursively, determining the set of reachable terminal histories from some history h, from the terminal histories which are reachable from the children of h.

Theorem 1. *Reachability finds the exact set of terminal history sets which are reachable from the root history.*

Proof. **By structural induction on the tree**

Let $P(n)$ be the proposition that is the conjunction, for all h of height n, of:

1) $Z_h(\sigma_1, \sigma_2) \in S_h \iff (\sigma_1, \sigma_2)$ is a feasible strategy pair from the subgame rooted at h

2) $\forall (\sigma_1, \sigma_2) \in \Pi_{1,h} \times BR(\sigma_1)_h$: $\pi_h(\sigma_1, \sigma_2) = pi_h(\sigma_1, \sigma_2)$

Note that $\Pi_{1,h} \times BR(\sigma_1)_h$ *is simply the set of feasible strategy pairs in the subgame* G_h.

Base Case ($|h| = 0$):

There is only a single strategy pair within the subgame rooted at h - the strategy pair which takes no action. Thus, this strategy pair is obviously feasible and the probability of terminating at h when playing it is 1. So, P(0) is true.

Inductive Case (Assume $P(k-1)$):

Let h be some history with $|h| = k$.

Algorithm 1 Reachability

function REACHABILITY $\{$

for $h \in H$

S_h will contain all of the reachable terminal history sets in the subgame rooted at h

if $|h| = 0$:

$S_h = \{h\}$

$pi_h(\sigma_1, \sigma_2)[h] = 1$

if $\rho(h) = p_1$:

Set $S_h = \cup_{c \in \chi(h)} S_c$

for each $c \in \chi(h)$ and $Z(\sigma_1, \sigma_2) \in S_c$:

$pi_h(\sigma_1, \sigma_2) = pi_c(\sigma_1, \sigma_2)$

if $\rho(h) = p_2$:

for each $c \in \chi(h)$, define :

$m(c) := \min_{Z_c(\sigma_1,\sigma_2) \in S_c} \mathbb{E}[\mu_2(\sigma_1, \sigma_2)]$

$s_h(c) := \max_{c \in \chi(h) \setminus \{c\}} m(c)$

for each $c \in \chi(h)$ and $Z_c(\sigma_1, \sigma_2) \in S_c$:

if $\mathbb{E}[\mu_2(\sigma_1, \sigma_2)] \geq s_h(c)$:

Add $Z_c(\sigma_1, \sigma_2)$ to S_h

Set $pi_h(\sigma_1, \sigma_2) = pi_c(\sigma_1, \sigma_2)$

else:

Set $pi_h(\sigma_1, \sigma_2) = 0$

if $\rho(p_c)$:

Let $\mathcal{X} = \times_{c \in \chi(h)} S_c$

for all $Z_\times \in \mathcal{X}$:

(i.e.: $Z_\times = \{Z_{c_1}(\sigma_1^1, \sigma_2^1), Z_{c_2}(\sigma_1^2, \sigma_2^2), ..., Z_{c_{|\chi(h)|}}(\sigma_1^{|\chi(h)|}, \sigma_2^{|\chi(h)|})\}$*)*

$S_h \leftarrow \cup_{(Z_\times)} Z_{c_k}(\sigma_1^k, \sigma_2^k)$ *Add the union of all terminal history sets in* $Z_\times$ *to* S_h

for each $Z_{c_k}(\sigma_1^k, \sigma_2^k) \in Z_\times$:

$pi_h(\sigma_1, \sigma_2) = \mathcal{C}(a) * pi_c(\sigma_1, \sigma_2)$

where a is the action taken by p_c at h such that $ha = c_k$.

$\}$

If $\rho(h) = p_1$:

From each $c \in \chi(h)$, player 1 can force, by their choice of σ_1, any stratregy pair (σ_1^c, σ_2^c) which is feasible in the subgame rooted at c. This is by $P(k-1)$.

As player 1 is acting at h, they can cause the game to enter the subgame rooted at any of the children of h by their choice of action at h.

Therefore, player 1 can cause, for any of these feasible strategy pairs (σ_1^c, σ_2^c), to be feasible in the subgame rooted at h.

Likewise, as player 1 only acts with pure strategies, $\pi_h(\sigma_1^c, \sigma_2^c) = \pi_c((\sigma_1^c, \sigma_2^c)$. As

$$\pi_c(\sigma_1^c, \sigma_2^c) = pi_c(\sigma_1^c, \sigma_2^c)$$

by $P(k-1)$,

$$pi_h(\sigma_1^c, \sigma_2^c) = pi_c(\sigma_1^c, \sigma_2^c)$$

also.

If $\rho(h) = p_2$:

player 2 must choose some action at h which comprises a best response to the strategy commitment of p_1 in the subgame G_h. That is, given p_1's strategy σ_1, it must choose a strategy which satisfies:

$$\sigma_2 = \arg\max_{\sigma_2' \in \Pi_{2,h}} \mathbb{E}[\mu_2(\sigma_1, \sigma_2')]$$

We know, by $P(k-1)$, the set of feasible strategy pairs in the subgame beginning at each $c \in \chi(h)$.For a specific child c, this set is equal to S_c.

A feasible strategy pair (σ_1, σ_2) in G_h must follow the actions of some feasible strategy pair in the subgame G_c rooted at the child chosen from h. Otherwise, σ_2 would not be a best response to σ_1 in G_c. Then, as (σ_1, σ_2) enters G_c, σ_2 wouldn't be a best response in G_h.

Fix some $c^0 \in \chi(h)$, and for each $c^i \in \chi(h) - \{c^0\}$, fix some arbitrary terminal history set $Z(\sigma_1^{c_i}, \sigma_2^{c_i}) \in S_{c_i}$. Then, for some $Z(\sigma_1^{c_0}, \sigma_2^{c_0}) \in S_{c_0}$ we will determine whether

$(\sigma_1^{c'}, \sigma_2^{c'})$ is feasible. This will then give us a rule to determine if we can commit from each child of h so that $Z(\sigma_1, \sigma_2) \in \cup_{c \in \chi(h)} S_c$ is reachable, and therefore (σ_1, σ_2) is feasible.

As $\rho(h) = p_2$, p_2 must choose an action which causes the game to enter a $c \in \chi(h)$. In order to be part of a best response strategy, this action must satisfy:

$$\arg\max_{c^i \in \chi(h)} \mathbb{E}[\mu_2(\sigma_1^{c_i}, \sigma_2^{c_i})]$$

Intuitively, p_2 must choose to enter a child history under which p_2's expected reward is maximal when playing a best response to the strategy committed to by p_1. And so, Z($\sigma_1^{c_0}, \sigma_2^{c_0}$) $\in S_{c_0}$ is reachable if there is some commitment $\sigma_1^{c_i}$ at every other child $c_i \in \chi(h) \backslash \{c_0\}$ by p_1 and subsequent best response $\sigma_2^{c_i}$ of p_2 for which $\mathbb{E}[\mu_2(\sigma_1^{c_0}, \sigma_2^{c_0})] \geq \mathbb{E}[\mu_2(\sigma_1^{c_i}, \sigma_2^{c_i})]$

From this, it is obvious that a $Z(\sigma_1, \sigma_2) \in S_c$ is reachable iff, for each

$$c_i \in \chi(h) \backslash \{c_0\}$$

and

$$(\sigma_1^{c_i,min}, \sigma_2^{c_i,min}) = \arg\min_{(\sigma_1^{c_i}, \sigma_2^{c_i}) \in S_{c_i}} \mathbb{E}[\mu_2(\sigma_1^{c_i}, \sigma_2^{c_i})]$$

we have that

$$\mathbb{E}[\mu_2(\sigma_1, \sigma_2)] \geq \mathbb{E}[\mu_2(\sigma_1^{c_i,min}, \sigma_2^{c_i,min})]$$

Of course, $\mathbb{E}[\mu_2(\sigma_1^{c_0}, \sigma_2^{c_0})]$ is greater than all $\mathbb{E}[\mu_2(\sigma_1^{c_i,min}, \sigma_2^{c_i,min})]$ iff it is greater than the maximum of these minimums:

$$(\sigma_1^{c_i,maxmin}, \sigma_2^{c_i,maxmin}) = \arg\max_{c_i \in \chi(h) \backslash \{c_0\}} \mathbb{E}[\mu_2(\sigma_1^{c_i,min}, \sigma_2^{c_i,min})]$$

.

But, $s_h(c) = \mathbb{E}[\mu_2(\sigma_1^{c_i,maxmin}, \sigma_2^{c_i,maxmin})]$. So, a strategy pair in S_{c_0} is feasible iff its expected reward to p_2 is at least as great as $s_h(c_0)$.

As p_2 can only act in pure strategies, if $Z(\sigma_1^{c_0}, \sigma_2^{c_0})$ is reachable, then $pi_h(\sigma_1^{c_0}, \sigma_2^{c_0}) = pi_c(\sigma_1^{c_0}, \sigma_2^{c_0})$.

If $\rho(h) = p_c$:

p_c will choose to enter each $ha = c \in \chi(h)$ with positive probability $\mathcal{C}(a)$. Thus, for each combination of feasible strategy pairs played from the children of h, the union of terminal histories reachable from any of these pairs is reachable from h.

More formally, for each cross product $(Z_{c_1}(\sigma_1^{c_1}, \sigma_2^{c_1}), Z_{c_2}(\sigma_1^{c_2}, \sigma_2^{c_2}), ...) \in \times_{c \in \chi(h)} S_c$, the union $\cup_{c \in \chi(h)} Z_c(\sigma_1^c, \sigma_2^c)$ is (probabilistically) reachable from h and so the corresponding strategy pair is feasible.

Similarly, each $Z_c(\sigma_1^c, \sigma_2^c)$ is reachable with probability $\mathcal{C}(a)$; where a is the action by p_c such that $ha = c$.

□

3.5 Strategy Pruning and Regret - Preliminaries

The complexity of **Reachability** is roughly proportional to the number of feasible strategy pairs (which explodes in number with the addition of each chance history). One obvious solution, then, is to limit the number of strategy pairs which we consider.

The approximation algorithm will "prune" strategy pairs at each history, maintaining enough to guarantee the desired approximation bound while removing enough to reduce the computational complexity to polynomial in both the number of histories and ϵ.

In order to determine which strategy pairs to consider, we need some measure of solution quality which allows us to measure the loss of removing a strategy pair from consideration. For this, we will utilize *regret*. Let Π'_1 be the set of player 1's strategies which we are considering, BR' be player 2's un-pruned best responses, and G' be the *pruned game* resulting from the strategy pruning. Then,

$$Regret(G \to G') := \underset{\sigma_1^{'} \in \Pi_1; \sigma_2^{'} \in BR(\sigma_1)}{\arg\max} \mu_1(\sigma_1^{'}, \sigma_2^{'}) - \underset{\sigma_1^{"} \in \Pi_1; \sigma_2^{"} \in BR(\sigma_1^{"})}{\arg\max} \mu_1(\sigma_1^{"}, \sigma_2^{"})$$

3.5.1 Bounding the Regret of a Pruned Strategy Set

Before providing the regret bounding algorithm, we must understand a few preliminary results on which the regret bound depends. These preliminaries will show which strategy pairs can and cannot be safely pruned .

Before that we will need to define a few more terms. First, we will utilize the idea of *dominated* and *dominating* strategy pairs.

dominated/dominating strategy pair: A strategy pair (σ_1, σ_2) is *dominated* by another strategy pair (σ_1', σ_2') iff there is a history h at which one of the following holds:

$$\mu_1(Z_h(\sigma_1, \sigma_2)) \leq \mu_1(Z_h(\sigma_1', \sigma_2')) \text{ and } \mu_2(Z_h\sigma_1, \sigma_2)) < \mu_2(Z_h(\sigma_1', \sigma_2'))$$

$$\mu_1(Z_h(\sigma_1, \sigma_2)) < \mu_1(Z_h(\sigma_1', \sigma_2')) \text{ and } \mu_2(Z_h(\sigma_1, \sigma_2)) \leq \mu_2(Z_h(\sigma_1', \sigma_2'))$$

Then, we say that (σ_1, σ_2) is dominated by (σ_1', σ_2').

Intuitively, both player 1 and player 2 prefer $Z_h(\sigma_1', \sigma_2')$ to $Z_h(\sigma_1, \sigma_2)$ and so each would choose to play according to the strategy pair (σ_1', a_2') from h over (σ_1, σ_2). Because of this, we don't need to consider a dominated strategy pair as it will never be chosen.

With this, we can define our final terms. For each history h:

$$s(h) := \underset{c \in \chi(h)}{\arg\max}\, s(c)$$

$s^{-1}(h)$ is the non-dominated strategy pair associated with s(h)

$Z(s^{-1}(h))$ is the terminal history set associated with $s^{-1}(h)$

with this we can define the exact class of strategy pairs which can be safely pruned:

Lemma 1. *There is a game G which satisfies the following: if a strategy pair, which is $s^{-1}(h)$ for some history h in G with $\rho(h) = p_2$ or $\rho(h) = p_c$, is removed to create the pruned game G', then **Reachability** on G' will choose an equilibrium strategy pair which is non-feasible in G.*

Proof. We first need to establish that, for each strategy pair (σ_1, σ_2), $\mathbb{E}[\mu_1(\sigma_1, \sigma_2)] \geq s(h) \iff \mathbb{E}[\mu_1(\sigma_1, \sigma_2)] \geq s_h(c)$.

For some history h and $c \in \chi(h)$, a strategy pair $Z(\sigma_1, \sigma_2) \in S_c$ is reachable at h iff $\mathbb{E}[\mu_1(\sigma_1, \sigma_2)] \geq s_h(c)$. $s(h) = max_{c \in \chi(h)} s_h(c)$, but this is equal to $s_h(c)$ unless $s^{-1}(h) = (\sigma_1, \sigma_2)$. In this case, though, all S_c would be reachable as the $Z(\sigma_1', \sigma_2') \in S_c$ with minimal reward to player 2 is $s^{-1}(h)$. Thus, $\mathbb{E}[\mu_1(\sigma_1, \sigma_2)] \geq s(h) \iff \mu_2(\sigma_1, \sigma_2) \geq s_h(c)$.

Now, assume that h is the root history of G and has two children: c_1 and c_2. S_{c_1} and S_{c_2} both contain a single terminal history set; respectively $Z_{c_1}(\sigma_1^{c_1}, \sigma_2^{c_1})$ and $Z_{c_2}(\sigma_1^{c_2}, \sigma_2^{c_2})$. Also, $\mu_1(\sigma_1^{c_1}, \sigma_2^{c_1}) > \mu_1(\sigma_1^{c_2}, \sigma_2^{c_2})$ and $\mu_2(\sigma_1^{c_1}, \sigma_2^{c_1}) < \mu_2(\sigma_1^{c_2}, \sigma_2^{c_2})$.

In the original game then, $(\sigma_1^{c_1}, \sigma_2^{c_1})$ is not feasible. If $(\sigma_1^{c_2}, \sigma_2^{c_2})$ is pruned though, $(\sigma_1^{c_1}, \sigma_2^{c_1})$ becomes feasible and will be the feasible strategy pair which maximizes the reward of player 1. Thus, **Reachability** will choose $(\sigma_1^{c_1}, \sigma_2^{c_1})$ as the equilibrium strategy pair. □

Lemma 2. *For all games $G \in \mathcal{G}$, if G is pruned of some strategy pair (σ_1, σ_2) which is not $s^{-1}(h)$ for any $h \in H$ to create the pruned game G', then the set of feasible strategy pairs in G' is exactly equivalent to the set of feasible strategy pairs besides (σ_1, σ_2) in G.*

Proof. From the proof of theorem 1, we have that $\mathbb{E}[\mu_1(\sigma_1, \sigma_2)] \geq s(h) \iff \mathbb{E}[\mu_1(\sigma_1, \sigma_2)] \geq s_h(c)$.

With this, it is obvious that the removal of (σ_1, σ_2) from G will not alter the set of feasible strategy pairs in G'. As we have not removed $s^{-1}(h_\emptyset)$, $s(h_\emptyset) = s(h'_\emptyset)$; where $h'_\emptyset$ is the root history of G'. A strategy pair (σ_1^*, σ_2^*) is reachable from some history h iff $\mathbb{E}[\mu_2(\sigma_1^*, \sigma_2^*)] \geq s(h)$. As $s(h_\emptyset) = s(h'_\emptyset)$, $\mathbb{E}[\mu_2(\sigma_1^*, \sigma_2^*)] \geq s(h_\emptyset) \iff \mathbb{E}[\mu_2(\sigma_1^*, \sigma_2^*)] \geq s(h'_\emptyset)$. So, the set of feasible strategy pairs, except (σ_1, σ_2), is equivalent between G and G' □

Note that this allows us to design a pruning strategy of arbitrarily many removals for which **Reachability** will still correctly compute which non-pruned strategy pairs are feasible. This pruning strategy must simply ensure that all removals are in accordance with lemma 2. The regret-bounding algorithm, utilizes this to simultaneously prune strategy pairs while running **Reachability** in such a way as to find an additively-approximate pure Stackelberg Equilibrium while reducing **Reachability** to poly-time in the inverse of the regret and linear in the size of H.

3.5.2 Ordering of Feasible and non-dominated Strategy Pairs

Lastly, we must define an ordering of feasible and non-dominated strategy pairs. For this ordering, we will simply use player 1's preference relation over these strategy pairs; where of course player 1 prefers a strategy pair (σ_1, σ_2) to a strategy pair (σ_1', σ_2') iff $\mu_1(\sigma_1, \sigma_2) > \mu_1(\sigma_1', \sigma_2')$. This creates a unique ordering on these strategy pairs except for when both $\mu_1(\sigma_1, \sigma_2) = \mu_1(\sigma_1', \sigma_2')$ and $\mu_2(\sigma_1, \sigma_2) = \mu_2(\sigma_1', \sigma_2')$. As there is no meaningful distinction between the two in relation to the achievable maximum reward to player 1, we can simply prune one of the two arbitrarily with 0 regret, and so we will treat one of the strategy pairs as dominated in this situation. For some history h, let $(\sigma_1, \sigma_2)_h^i$ be the i-th strategy pair in the ordering of feasible and non-dominating strategy pairs at h.

Note that this ordering contains exactly the set of strategy pairs which are relevant to finding a Stackelberg Equilibrium in the game. Dominated strategies will always have some dominating strategy which both players 1 and 2 prefer, and so will never be chosen as the equilibrium strategy. Likewise, non-feasible strategies do not satisfy the requirements of a Stackelberg Equilibrium, and so cannot be chosen as the equilibrium strategy.

3.6 Regret-Bounding Algorithm

The regret-bounding algorithm augments Reachability so that only enough strategy pairs are considered to keep $Regret$ within the desired bound: $0 < \epsilon < 1$.

As in the exact algorithm, We will be running Reachability on our game G from the terminal histories to the root of the tree. But, now we will also be pruning strategies pairs at each history, to create a pruned game G', so that the size of S_h is no more than $2 \cdot \lceil \frac{1}{\epsilon} \rceil + 1$ at any history h. Specifically, we will keep track of $2 \cdot \lceil \frac{1}{\epsilon} \rceil$ strategy pairs and $s^{-1}(h)$ in such a way as to bound $Regret(G_h, G'_h) \leq \epsilon$ for all subgames.7

These $2 \cdot \lceil \frac{1}{\epsilon} \rceil$ will be split into two groups of $\lceil \frac{1}{\epsilon} \rceil$. One of these groups will consist of strategy pairs (σ_1, σ_2) which, for each $k \in [0, \lceil \frac{1}{\epsilon} \rceil - 1]$ which have $\mu_1(\sigma_1, \sigma_2) \geq (1 - k * \epsilon)$ and have μ_1 "close enough" to $(1 - k * \epsilon)$. The second group is defined similarly, except that $\mu_1(\sigma_1, \sigma_2) \leq (1 - k * \epsilon)$. Let the k-th of each set be denoted, respectively, by $S_h(k, high)$ or $S_h(k, low)$.

Algorithm 2 ϵ-Reachability

function ϵ-REACHABILITY(:)

Throughout, assume that pi_h is set for all un-pruned strategy pairs as would be done in Reachability.

for each $h \in H$:

if $|h| = 0$:

for all $k \in [0, \lceil \frac{1}{\epsilon} \rceil - 1]$:

Let (σ_1, σ_2) be the (only) strategy pair in the subgame rooted at the terminal history h. Set:

$S_h(k, low) = Z(\sigma_1, \sigma_2)$

$S_h(k, high) = Z(\sigma_1, \sigma_2)$

$s(h) = \mu_2(\sigma_1, \sigma_2)$,

Now, assume that $|h| > 0$

if $\rho(h) = p_1$:

$s(h) = \max_{c \in c} s(C)$

for increasing k from 0 to $\frac{1}{\epsilon} - 1$: $S_h(k, high) = setS_h(k, \text{“}high\text{”})$ // Defined below $S_h(k, low) = setS_h(k, \text{“}low\text{”})$

if $\rho(h) = p_2$:

Determine, for what terminal history sets in S_c for each $c \in \chi(h)$ as is done in Reachability.

for each $c \in \chi(h)$:

for each $S_c(k, low)$ and $S_c(k, high)$ found infeasible by **Reachability** :

Set $S_c(k, \bullet)$ to be the strategy pair $(\sigma_1', \sigma_2') \in S_c$ which is reachable from h and has maximum $\mu_1(\sigma_1', \sigma_2')$.

With these updated S_c, proceed as if $\rho(h) = p_1$.

if $\rho(h) = p_c$:

Assume that $|h| = 2$. If not, expand h into a series of histories each with two children.

With pr as the probability of the chance player selecting c_1, set $s(h) = pr * s(c_1) + (1 - pr) * s(c_2)$

We will perform **Reachability** as in the exact solution, but select each strategy pair as was done for the $\rho(h) = p_1$ case. In other words, we determine what each $S_h(k, low)$ and $S_h(k, high)$ will be according to the rules in the $\rho(h) = p_1$ case, but consider all strategies in the cross product of S_{c_1} and S_{c_2}.

function $setS_h(k, type)$:

if $type ==$ "high" :
In order of preference, return one of the following:

1): Between all $c_i \in \chi(h)$, the $S_{c_i}(k, high)$ that is non-dominated and has $\mu_1(S_{c_1}(k, high) \geq (1 - k * \epsilon)$. If multiple children satisfy this, then choose the $S_{c_i}(k, high)$ with minimal $\mu_1(S_{c_i}(k, high))$.
2): Between all $c_i \in \chi(h)$, the $S_{c_i}(k, high)$ which is non-dominated. If all children satisfy this, then choose the one with maximum $\mu_1(S_{c_i}(k, high))$.
3) If $S_{c_i}(k, high)$ are dominated for all children, then set $S_h(k, high) = S_h(k - 1, low)$ *(which corresponds to the dominating strategy pair)*.

if $type ==$ "low" :
In order of preference, return one of the following:

1) Between all $c_i \in \chi(h)$ and $S_h(k, high)$, the $Z(\sigma_1, \sigma_2)$ which is non-dominated and has $\mu_1(\sigma_1, \sigma_2) \leq (1 - k * \epsilon)$. If all satisfy this, then choose the one with maximum μ_1.
2) Between all $c_i \in \chi(h)$, the $S_{c_i}(k, high)$ for which c_i is non-dominated. If both children satisfy this, then choose the one with minimal $|\mu_1(S_{c_i}(k, high)) - (1 - k * \epsilon)|$.
3) If, for all children $c \in \chi(h)$, $S_c(k, low)$ is dominated, then set $S_h(k, low) = S_h(k, high)$ *(which corresponds to dominating strategy pair)*.

Theorem 2. *If OPT is the reward to player 1 under an optimal strategy pair, then ϵ-**Reachability** computes a feasible strategy pair (σ_1, σ_2) with $\mu_1(\sigma_1, \sigma_2) \geq OPT - \epsilon$.*

Proof. Throughout we will write (σ') as shorthand for the strategy pair (σ'_1, σ'_2).

for each subgame G_h, we will show that the following invariant holds:

Invariant:

For each feasible, non-dominated strategy pair (σ') which is pruned in G'_h, for some $k \in \lceil \frac{1}{\epsilon} \rceil$ either:

1) $(1 - (k+1) * \epsilon) \leq \mu_1(S_h(k, low)) \leq (1 - k * \epsilon) \leq \mu_1(\sigma')$

and $\mu_1(\sigma') - \mu_1(S_h(k, low)) \leq \epsilon$

2) $(1 - k * \epsilon) \leq \mu_1(S_h(k, high)) \leq \mu_1(\sigma') \leq (1 - (k-1) * \epsilon)$

Let (σ') be a feasible, non-dominated strategy in G_c for some $c \in \chi(h)$ which is pruned in G'_c. Let $(\sigma")$ be a feasible, non-dominated strategy in G_h and G_c for some $c \in \chi(h)$ but which is pruned in G'_h.

If $\rho(h) = p_1$:

For (σ'):

If (σ') satisfies **1** in G'_c:

As there is some $(1-k*\epsilon)_c(k, low) \leq (1-(k-1)*\epsilon)$, it will be that $S_h(k, low)$ is chosen such that $\mu_1(S_c(k, low)) \leq S_h(k, low) \leq (1-(k-1)*\epsilon)$ and $\mu_1(\sigma') - \mu_1(S_h(k, low) \leq \epsilon$.

If possible, $S_h(k, low)$ is chosen from some child $c_i \in \chi(h)$ to be the $S_{c_i}(k, low)$ which satisfies:

$$S_h(k, low) := \min_{c_i \in \chi(h)} (1 - k * \epsilon) - \mu_1(S_{c_i}(k, low))$$

$$\textit{s.t. } \mu_1(S_h(k, low)) \leq (1 - k * \epsilon)$$

As $\mu_1(S_c(k, low)) \leq (1 - k * \epsilon)$, there is a child which satisfies the constraint. So, we will have $S_h(k, low)$ chosen such that:

$$\mu_1(S_c(k, low)) \leq \mu_1(S_h(k, low)) \leq (1 - k) * \epsilon)$$

As $\mu_1(S_c(k, low)) \leq \mu_1(S_h(k, low)) \leq (1-k*\epsilon) \leq \mu_1(\sigma')$ and $\mu_1(\sigma') - \mu_1(S_c(k, low)) \leq \epsilon$, we will have $\mu_1(\sigma') - \mu_1(S_h(k, low)) \leq \epsilon$. Thus, (σ') satisfies **1** in G'_h.

If (σ') satisfies **2** in G'_c:

As there is some $(1 - k * \epsilon) \leq S_c(k, high) \leq \mu_1(\sigma') \leq (1 - (k - 1) * \epsilon)$, $S_h(k, high)$ will be chosen so that $(1 - k * \epsilon) \leq S_h(k, high) \leq \mu_1(\sigma') \leq (1 - (k - 1) * \epsilon)$.

If possible, $S_h(k, high)$ is chosen from some child $c_i \in \chi(h)$ to be the $S_{c_i}(k, low)$ which satisfies:

$$S_h(k, high) := \min_{c_i \in \chi(h)} \mu_1(S_{c_i}(k, high)) - (1 - k * \epsilon)$$

$$\textit{s.t.}\ (1 - k * \epsilon) \leq \mu_1(S_h(k, low))$$

As $(1 - k * \epsilon) \leq \mu_1(S_c(k, high))$, we have some c_i that satisfies the constraint. So, we will have $S_h(k, high)$ chosen such that:

$$(1 - k * \epsilon) \leq S_h(k, high) \leq S_c(k, high) \leq \mu_1(\sigma') \leq (1 - (k - 1) * \epsilon)$$

So, **2** holds for (σ').

For $(\sigma")$:

Without loss of generality, assume $(\sigma")$ *is removed from the* S_c *for some specific child* $c \in \chi(h)$ *and that* k *is chosen so that* $(1 - k * \epsilon) \leq \mu_1(\sigma") \leq (1 - (k - 1) * \epsilon)$.

If $(\sigma") = S_c(k, high)$:

As shown in the argument for a pruned (σ') satisfying **2**, if $S_c(k, high)$ is not chosen as $S_h(k, high)$, then some other strategy pair must be chosen such that

$$(1 - k * \epsilon) \leq S_h(k, high) \leq S_c(k, high) \leq (1 - (k - 1) * \epsilon)$$

Thus $(\sigma")$ satisfies **2** in G'_h.

If $(\sigma") = S_c(k, low)$:

As shown in the argument for a pruned (σ') satisfying **2**, if some $(\sigma")$ is removed then it must be that

$$(1 - k * \epsilon) \leq S_h(k + 1, high) \leq S_c(k, low) \leq (1 - (k - 1) * \epsilon)$$

Thus, $(\sigma")$ satisfies **2**.

If $\rho(h) = p_2$:

Note that we do not need to consider $(\sigma")$ *separately from that* $\rho(h) = p_1$ *case, as* $(\sigma")$ *cannot both satisfy it is defined to be feasible at* G_h.

Assume that $S_c(k, \cdot)$ is the terminal history set which satisfies the invariant for (σ') in G_c. We need to ensure that $S_c(k, \cdot)$ cannot become infeasible while (σ') remains feasible.

$\mu_2(\sigma') \leq \mu_2(S_c(k, \cdot)$ as both are non-dominated and $\mu_1(S_c(k, \cdot) \leq \mu_1(\sigma')$. A strategy pair (σ^*) remains feasible unless $\mu_2(\sigma^*) < s_h(c)$. If $\mu_2(S_c(k, \cdot) < s_h(c)$ then $\mu_2(\sigma') < s_h(c)$. Thus, $S_c(k, \cdot)$ cannot become infeasible while (σ') remains feasible.

After removing infeasible strategy pairs, both **Reachability** and ϵ**-Reachability** proceed identically to the case where $\rho(h) = p_1$. Thus, as the invariant holds during the additional beginning steps of the $\rho(h) = p_2$ case, it will hold for all remaining steps of the $\rho(h) = p_2$ case.

If $\rho(h) = p_c$:

For (σ'):

We assume that h has at most two children, c_1 and c_2.

Let (σ'_{c_1}) and (σ'_{c_2}) be the sub-strategy pairs played from each child of h in (σ') (*note that at least one of these must also be pruned in their respective subgame*). By the invariant, there will be some $(\sigma_{c_1}) \in S_{c_1}$ with $\mu_1(\sigma_{c_1}) \leq \mu_1(\sigma'_{c_1})$ and $\mu(\sigma_{c_1}) - \mu_1(\sigma'_{c_1}) \leq \epsilon$ and some $(\sigma_{c_2}) \in S_{c_2}$ with $\mu_1(\sigma_{c_2}) \leq \mu_1(\sigma'_{c_2})$ and $\mu(\sigma_{c_2}) - \mu_1(\sigma'_{c_2}) \leq \epsilon$. Let (σ) be the strategy pair in G_h created by following (σ_{c_1}) from c_1 and (σ_{c_2}) from c_2. If pr is the probability that the chance player selects c_1, then:

$$0 \leq \mu_1(\sigma) - \mu_1(\sigma') =$$

$$pr\Big(\mu_1(\sigma'_{c_1}) - \mu_1(\sigma_{c_1})\Big) + (1 - pr)\Big(\mu_1(\sigma'_{c_2}) - \mu_1(\sigma_{c_2})\Big)$$

$$\leq pr * \epsilon + (1 - pr) * \epsilon = \epsilon$$

Thus, $0 \leq \mu_1(\sigma'_1) - \mu_1(\sigma_1) \leq \epsilon$.

If there is some q such that $(1 - q * \epsilon) \leq \mu_1(\sigma) \leq \mu_1(\sigma') \leq (1 - (q-1) * \epsilon)$, then as with the case where $\rho(h) = p_1$, $S_h(q, high)$ will be chosen such that

$$(1 - q * \epsilon) \leq S_h(q, high) \leq \mu_1(\sigma)$$

So, $(1 - q * \epsilon) \leq S_h(k, high) \leq \mu_1(\sigma') \leq (1 - (q-1) * \epsilon)$ and **2** of the invariant holds. But it may also be that there is no q satisfying the above but only a q such that $(1 - (q+1) * \epsilon) \leq \mu_1(\sigma) \leq (1 - q * \epsilon) \leq \mu_1(\sigma') \leq (1 - (q-1) * \epsilon)$. In this case, though, $S_h(k, low)$ will be chosen such that:

$$(1 - (q+1) * \epsilon) \leq \mu_1(\sigma) \leq \mu_1(S_h(k, low) \leq (1 - q * \epsilon) \leq \mu_1(\sigma')$$

As $\mu_1(\sigma') - \mu_1(\sigma) \leq \epsilon$, it will also be that $0 \leq \mu_1(\sigma') - \mu_1(S_h(k, low)) \leq \epsilon$ and so (σ') satisfies **1** of the invariant.

For $(\sigma")$:

Let $(\sigma"_{c_1}) \in S_{c_1}$ and $(\sigma"_{c_2}) \in S_{c_2}$ be the sub-strategy pairs which are played from each child of h in the strategy pair $(\sigma")$ and let k be such that $(1 - k * \epsilon) \leq \mu_1(\sigma") \leq (1 - (k-1) * \epsilon)$. As $(\sigma")$ was not chosen as $S_h(k, high)$, it will be chosen such that:

$$(1 - k * \epsilon) \leq \mu_1(S_h(k, high)) \leq \mu_1(\sigma") \leq (1 - (k-1) * \epsilon)$$

So, $(\sigma")$ satisfies **2** of the invariant.

□

Lemma 3. *ϵ-Reachability runs in $O(b\epsilon^{-2}|V|)$; where b is the maximum branching factor of the game tree and $|V|$ is the number of histories.*

Proof. We will show that, for any history h, the time complexity of computing S_h is no greater than $O(b\epsilon^{-2})$.

Note that S_h will include $\frac{1}{\epsilon}$ terminal history sets - possible including duplicates.

If $|h| = 0$:

All we must do is fill in the table with the rewards for players 1 and 2 into the $\frac{1}{\epsilon}$ spots. This can obviously be done in $O(\frac{1}{\epsilon})$ steps.

If $\rho(h) = p_1$:

For each $S_h(k, low)$ and $S_h(k, high)$, we consider no more than $4b$ strategy pairs from the b children of h. As there are a multiple of $\frac{1}{\epsilon}$ strategy pairs in S_h, this will take $O(b\epsilon^{-1})$.

If $\rho(h) = p_2$:

The algorithm for determining S_h for these histories is identical to p_1, except for an additional step in which it is determined which strategy pairs have become infeasible and replace them in each S_c.

It will take no longer than $O(b * \epsilon^{-1})$ to compute $s(h)$.

Then, it will take no longer than $O(b * \epsilon^{-1})$ to determine the infeasible strategy pairs - by simply iterating through them.

Finally, it will take no longer than $O(b * \epsilon^{-1})$ to replace the newly-infeasible strategy pairs in each S_c.

As the additional step for $\rho(h) = p_1$ takes $O(b * \epsilon)$ and this is equivalent to the worst-case complexity when $\rho(h) = p_1$, the complexity is $O(b * \epsilon)$

If $\rho(h) = p_c$:

We assume for these histories that h has exactly 2 children. If the branching factor is some $b \geq 2$, we expand h into a series of histories histories with two children. This will be linearly larger than b. Thus, if we can compute S_h for our h in $O(f(n))$ where $f(n)$ grows at least linearly fast, then we can compute S_h in $O(b * f(n))$ even for h with more than 2 children.

Let c_1 and c_2 be the two children of h.

As $|h| = 2$, we will have ϵ^{-2} pairs of strategy pairs in the cross product of S_{c_1} and S_{c_2}. Each of these combinations will have some reward $\mu_1(\sigma)$, which, for some $k \in \lceil \frac{1}{\lfloor}\epsilon \rceil$ it will be $(1 - (k-1) * \epsilon) \leq \mu_1(\sigma) \leq (1 - k * \epsilon)$. If, for each of these σ, we compare it against the current value of $S_h(k-1, high)$ and $S_h(k, low)$ (where each start out empty), we can determine, for each such k, the strategy pair which bests fits in each. This will take $O(\epsilon^{-2})$. There may still be spots in S_h which are not filled. Iterating from $S_h(\frac{1}{\epsilon}, high)$ to $S_h(0, low)$ and then in the opposite direction, these can be filled by choosing the best strategy pair from the slot above or below each slot which was not filled during the previous step. This will take $O(\epsilon^{-1})$. Thus, S_h can be computed in $O(\epsilon^{-2})$ and for arbitrary $|h| = b$, S_h can be computed in $O(b\epsilon^{-2})$.

□

Chapter 4

Simulations

Simulations were performed to determine the running time of the ϵ-**Reachability** algorithm and compared against an $FPTAS$ (Fully Polynomial Time Appoximation Scheme)[3] for the task of approximating a pure Stackelberg Equilibrium in extensive-form games with a chance player. Running times are compared between ϵ-Reachability and the $FPTAS$ while varying a number of relevant parameters.

4.1 $FPTAS$ Algorithm for Pure Stackelberg Equilibrium in Extensive-Form Games

The NP-Hardness result of finding a pure Stackelberg Equilibrium in extensive-form games with chance nodes was shown in [17] by a reduction from $KNAPSACK$, and the $FPTAS$ is closely related to the classic approximation scheme for the $KNAPSACK$ problem.

The algorithm begins by scaling all of player 1's rewards, but not player 2's, by some factor D. Equivalently, it discretizes the payments into multiples of $\frac{1}{D}$. As with the approximation scheme for $KNAPSACK$, the approximation error of the $FPTAS$ is dependent on the size of D as well as the height of the game tree. For this reason, we must have $D \leq \frac{\epsilon}{|h_\emptyset|}$, where $|h_\emptyset|$ is the height of the tree, to achieve a result which is within an additive error of ϵ of the optimal solution.

After discretization, the algorithm works by creating, for each subtree T, a table Tbl_T with $n = \lceil \frac{|h_\emptyset|}{\epsilon} \rceil$ entries. For each $q \in \lceil \frac{|h_\emptyset|}{\epsilon} \rceil$, $Tbl_T[q]$ satisfies two guarantees:

1) the leader has a strategy from T which attains, under some best response by the follower $Tbl_T[q]$ reward to the follower and at least q reward to the leader.

2) No leader strategy from T can offer the follower strictly more that $Tbl_T[q]$ while the leader attains at least $q + |T|$ reward; where $|T|$ is the height of T.

As with ϵ-**Reachability**, we assume that chance nodes (in which the chance player acts) have exactly two children. If there are more children than this, we expand the chance node into multiple levels of chance nodes, each with exactly two children. Per node, this results in only a small increase in the size of the game tree.

Similar to ϵ-**Reachability**, the values of Tbl_T are defined recursively and relative to the subtree's root node type (player action, chance, or terminal).

With $h_\emptyset^T$ as the root of the subtree:

if $h_\emptyset^T$ is terminal ($|h_\emptyset^T| = 0$):

Let μ_1, μ_2 be the rewards to players 1 and 2. Then, for each $q \in [0, n]$:

$$Tbl_T[q] := \begin{cases} \mu_2 \text{ if } q \leq \mu_1 \\ -\infty \text{ otherwise} \end{cases}$$

Obviously, this can be computed in $O(1)$ at each relevant node.

If $\rho(h_\emptyset^T) = p_1$:

$$Tbl_T[q] := \max_{h_\emptyset^c \in c(h_\emptyset^T); i \geq k} Tbl_c[i]$$

This can be computed in $O(bn)$ at each relevant node, where b is the maximum branching factor of the tree.

If $\rho(h_\emptyset^T) = p_2$:

For each child $h^c_\emptyset \in c(h)$ and each $q \in [0, n]$, define:

$$Tbl_c[q] \downarrow_\mu := \begin{cases} Tbl_c[q] \text{ if } q \geq \mu \\ -\infty \text{ otherwise} \end{cases}$$

$$m(h^c_\emptyset) := \max_{h^c_\emptyset \in c(h^T_\emptyset) \setminus \{h^c_\emptyset\}} \left(\min_{q \in [0,n]} Tbl_c[q] \right)$$

Then,

$$Tbl_T[q] := \max_{h^c_\emptyset \in c(h)} Tbl_c[q] \downarrow_{m(c)}$$

This can be completed in $O(bn)$ at each relevant node.

If $\rho(h^T_\emptyset) = p_c$:

If there are more than two children of a chance node, we expand the node into multiple levels of binary trees.

Let $h^{c_1}_\emptyset$ and $h^{c_2}_\emptyset$ be the root nodes of the two subtrees emanating each from a child of $h^T_\emptyset$. Then, for each $i, j \in [0, n]$, and with pr as the probability that the chance player chooses to enter $h^{c_1}_\emptyset$:

$$Tbl_T[q] := \max_{i,j} \{pr * Tbl_{c_1}(i) + (1 - pr) * Tbl_{c_2}[j]) \mid pr * i + (1 - pr) * j \geq k\}$$

This can be computed in $O(bn^2)$. As with ϵ-Reachability, expanding a node with more than 2 children into a series of nodes with 2 children results in only linearly more nodes. Thus, the expansion of the b children will take $O(b)$ and so the entire step will take $O(bn^2)$

Altogether, this results in a time complexity of $O(|V|bn^2) = O(|V|b(\frac{|h_\emptyset|}{\epsilon})^2)$, where $|V|$ is the number of nodes in the tree.

4.2 Simulation Parameters

Relevant factors to determine the $FPTAS$ run-time are $|V|$, b, $|h_\emptyset|$, and $\frac{1}{\epsilon}$. Simulatios will show how the altering of these variables (except b, which will always equal 2) affects the run-time of the algorithm. With the difference in complexity for determining Tbl at chance ($O(bn^2)$) and player action ($O(bn)$) nodes, we also need to consider the proportion of chance nodes in the game tree. For both algorithms, rewards are floating point numbers in the range $[0, 1]$; representing the normalized rewards assumed in the proof of correctness for both algorithms.

Likewise, ϵ**-Reachability**, which runs in $O(\epsilon^{-2}|V|)$, also has ϵ, $|V|$, and the proportion of chance nodes to player action nodes as relevant factors in run-time. It, though, does not depend on $|h_\emptyset|$. This will prove a major improvement on $FPTAS$ for tall trees.

Both algorithms are implemented in Python 3. Simulations are run on a standard Google Cloud *n1-highmem-8 (with 8 vCPUs and 52Gb Memory)* VM instance. All game trees are full, binary trees. This simplification is used because all trees can be expanded into binary trees and it is simpler to determine the size of and height of a binary tree. Simulation run-times are all averaged over 20 iterations.

Each node h is assigned as either a chance or player action node according to a pre-set probability, pr_c. Player action nodes are then set between $\rho(h) = p_1$ and $\rho(h) = p_2$ with equal probability.

In the next sections, we will see simulations testing the run-times of the two algorithms under varying combinations of the above parameters, as well as comparisons between the two when appropriate. The chapter will conclude with a comparative discussion of the two approximation schemes in terms of their suitability for approximating Stackelberg Equilibria in very large extensive-form games.

4.3 Simulation Results

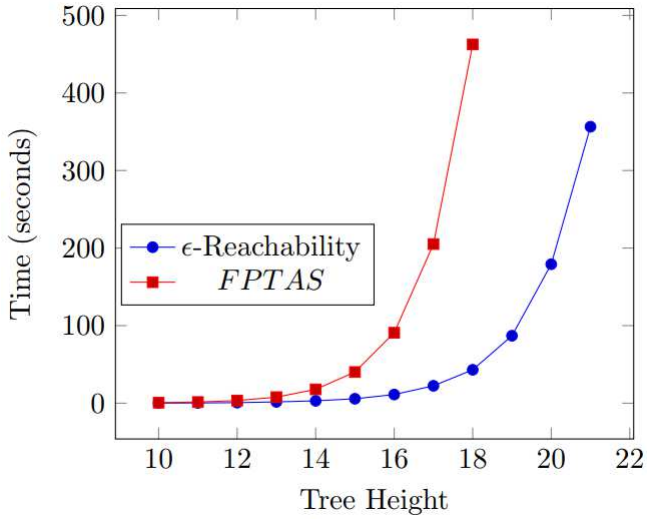
When each parameter is fixed, they will usually be set to "base" values for ease of comparison. For each factor this is:

- $|V| = 2^{12} - 1$ - *As game trees are full and binary,* $|V| = 2^{|h_\emptyset|+2} - 1$
- $|h_\emptyset| = 10$
- $b = 2$ - *for all non-terminal nodes*
- $\epsilon = 0.1$
- $pr_c = 0.5$

These were all chosen because they are small enough that simulations run quickly while being large enough to see variations in the other variables.

4.3.1 Tree Size/Height Comparisons

Figure 4.1 Comparisons with varying tree height ($|h_\emptyset|$)



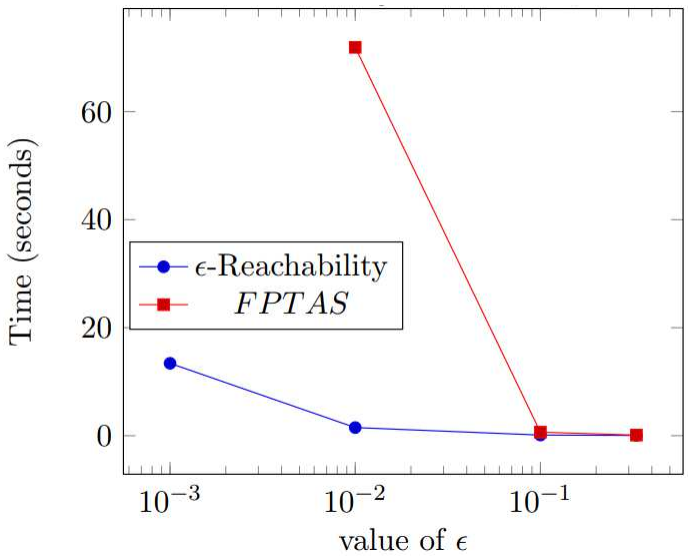
The above graph shows the running time of the $FPTAS$ (red blocks) and ϵ-Reachability (blue circles) for increasingly values of $|h_\emptyset|$.

The $FPTAS$ is quadratic in the height of the game tree, and this shows in the run times. While both algorithms have run times that roughly double as each level is added to the (full) binary tree, the $FPTAS$ run time also grows quadratically in the height of tree.

It is difficult to see from the graph, but the the running time of the algorithm for $|h_\emptyset| = 10$ is almost four times longer for the $FPTAS$, at 0.4 seconds, than ϵ-Reachability, at 0.13 seconds.

4.3.2 ϵ Comparisons

Figure 4.2 Comparisons with varying ϵ



There is a clear difference between the ϵ-Reachabiility and $FPTAS$ run time as ϵ shrinks, the growth rate of the $FPTAS$'s run-time substantially outpaces that of ϵ-Reachability for smaller values. It becomes more than three times as long at $\epsilon = 0.01$.

4.3.3 Pr_c Comparisons

Height of Game Tree	FPTAS		ε-Reachability	
	Pr_C: 0.1	Pr_C: 0.9	Pr_C: 0.1	Pr_C: 0.9
15	13.48s	67.01s	4.07s	8.02s
16	29.39s	152.19s	8.41s	16.50s
17	65.24s	343.94s	16.86s	30.98s
18	142.61s	774.16s	35.65s	59.61s

Table 4.1 Pr_c comparisons

Tests were performed with values of 0.1 and 0.9 for pr_c. For ϵ-Reachability, the time complexity on non-chance action nodes is $O(b\epsilon^{-2})$ for chance nodes and $O(b\epsilon^{-1})$ for player action nodes. $FPTAS$ is simlar, execpt with an additional factor of $|h_\emptyset|$ to make the complexity $O(b\frac{|h_\emptyset|}{\epsilon})$ on non-chance player nodes and $O(b(\frac{|h_\emptyset|}{\epsilon})^2)$ on chance nodes. This difference can clearly be seen in the run-times of the algorithm with varying values of pr_c - the probability that each node is a chance node. Both algorithms run on games with less chance nodes substantially faster than those with more chance nodes. $FPTAS$, though, grows much more quickly with pr_c. This growth rate increaes with the height of the tree.

4.3.4 Discussion

For all trees of height greater than 2, $FPTAS$ must maintain a substantially larger table than ϵ-Reachability. While the run-time of both algorithms are similar for each table entry, this larger table of $FPTAS$ makes it grow much more quickly than ϵ-Reachability. Even for trees of a relatively modest height of 10, it can be seen from simulations that $FPTAS$ runs substantially slower than ϵ-Reachability.

It appears that ϵ-Reachability is a possible solution to expanding the size of extensive-form games for which we can approximate pure Stackelberg Equilibria. It's run-time has a substantially less steep growth curve in ϵ^{-1}, even for a relatively modest tree height of 15. $FPTAS$ is also much slower even for moderate levels of $|h_\emptyset|$ and ϵ. This may also allow

it to degrade its approximation more gracefully for large games than would be possible with $FPTAS$.

Chapter 5

Conclusion

In this thesis ϵ-Reachabilitty was introduced as a new $OPT - \epsilon$ approximation algorithm for pure Stackelberg Equilibria in extensive-form games. It was shown that the algorithm was $O(|V|b\epsilon^{-2})$, where $|V|$ is the number of nodes in the game tree and b is the maximum branching factor. ϵ-Reachability was found to theoretically outperform the previous approximation algorithm $FPTAS$ in the worst case, having greater than linear complexity in only ϵ^{-1} while the previous algorithm was polynomial in both ϵ^{-1} and the height of the game tree $|h_\emptyset|$. In simulations, it was shown that the run-time of the approximation algorithm grows much more quickly with decreasing ϵ than ϵ-Reachability. This larger run-time was greatly exacerbated by growing values of $|h_\emptyset|$, and the inclusion of $|h_\emptyset|$ in the complexity of $FPTAS$ is likely responsible for the consistently longer run-time.

These results suggest that ϵ-Reachability is a strong alternative to $FPTAS$; especially on trees which are at least moderately tall. For these trees, the run-time of ϵ-Reachability increases more slowly in both $|h_\emptyset|$ and ϵ. This suggests that ϵ-Reachability may be quick enough for use on trees for which a combination of these parameters has made the previous approximation algorithm infeasible.

LIST OF REFERENCES

[1] Ishfaq Ahmad, Sanjay Ranka, and Samee Ullah Khan. Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–6. IEEE, 2008.

[2] Anastasios G Bakirtzis. Aumann-shapley transmission congestion pricing. *IEEE Power Engineering Review*, 21(3):67–69, 2001.

[3] Branislav Bošanskỳ, Simina Brânzei, Kristoffer Arnsfelt Hansen, Troels Bjerre Lund, and Peter Bro Miltersen. Computation of stackelberg equilibria of finite sequential games. *ACM Transactions on Economics and Computation (TEAC)*, 5(4):23, 2017.

[4] Branislav Bosansky and Jiri Cermak. Sequence-form algorithm for computing stackelberg equilibria in extensive-form games. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[5] Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. A game theoretic framework for incentives in p2p systems. *arXiv preprint cs/0310039*, 2003.

[6] Ioannis Caragiannis. Efficient coordination mechanisms for unrelated machine scheduling. *Algorithmica*, 66(3):512–540, 2013.

[7] Po-An Chen and Chi-Jen Lu. Playing stackelberg opinion optimization with randomized algorithms for combinatorial strategies. *arXiv preprint arXiv:1803.01792*, 2018.

[8] Constantinos Daskalakis. On the complexity of approximating a nash equilibrium. *ACM Transactions on Algorithms (TALG)*, 9(3):23, 2013.

[9] Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309, 2003.

[10] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, pages 2589–2595, 2015.

[11] Zuyuan Fang and Brahim Bensaou. Fair bandwidth sharing algorithms based on game theory frameworks for wireless ad hoc networks. In *IEEE infocom*, volume 2, pages 1284–1295. Citeseer, 2004.

[12] Sergiu Hart and David Schmeidler. Existence of correlated equilibria. *Mathematics of Operations Research*, 14(1):18–25, 1989.

[13] Nicole Immorlica, Li Erran Li, Vahab S Mirrokni, and Andreas S Schulz. Coordination mechanisms for selfish scheduling. *Theoretical computer science*, 410(17):1589–1598, 2009.

[14] Peter B Key and Derek R McAuley. Differential qos and pricing in networks: where flow control meets game theory. *IEE Proceedings-Software*, 146(1):39–43, 1999.

[15] Christian Kroer and Tuomas Sandholm. Extensive-form game abstraction with bounds. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 621–638. ACM, 2014.

[16] Yu-Kwong Kwok, ShanShan Song, and Kai Hwang. Selfish grid computing: game-theoretic modeling and nas performance results. In *2005 IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2005*. IEEE. The Journal's web site is located at http://ieeexplore. ieee. org/xpl/conhome. jsp? punumber= 1000093, 2005.

[17] Joshua Letchford and Vincent Conitzer. Computing optimal strategies to commit to in extensive-form games. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 83–92. ACM, 2010.

[18] Igal Milchtaich. Congestion games with player-specific payoff functions. *Games and economic behavior*, 13(1):111–124, 1996.

[19] Amitav Mukherjee and A Lee Swindlehurst. Jamming games in the mimo wiretap channel with an active eavesdropper. *IEEE Transactions on Signal Processing*, 61(1):82–91, 2013.

[20] Thanh Hong Nguyen, Jason Tsai, Albert Xin Jiang, Emma Bowring, Rajiv T Maheswaran, and Milind Tambe. Security games on social networks. In *AAAI Fall Symposium: Social Networks and Social Contagion*, 2012.

[21] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic behavior*, 35(1-2):166–196, 2001.

[22] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[23] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[24] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi. Iris-a tool for strategic security allocation in transportation networks. 2009.

[25] Kevin Waugh, David Schnizlein, Michael Bowling, and Duane Szafron. Abstraction pathologies in extensive games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 781–788. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[26] Liang Xiao, Caixia Xie, Tianhua Chen, Huaiyu Dai, and H Vincent Poor. A mobile offloading game against smart attacks. *IEEE Access*, 4:2281–2291, 2016.

[27] Guanghui Zhou, Zhongdong Xiao, Pingyu Jiang, and George Q Huang. A game-theoretic approach to generating optimal process plans of multiple jobs in networked manufacturing. *International Journal of Computer Integrated Manufacturing*, 23(12):1118–1132, 2010.